CS 3300: Object Oriented Programming using Java

Spring 2016

Course Project, Phase II

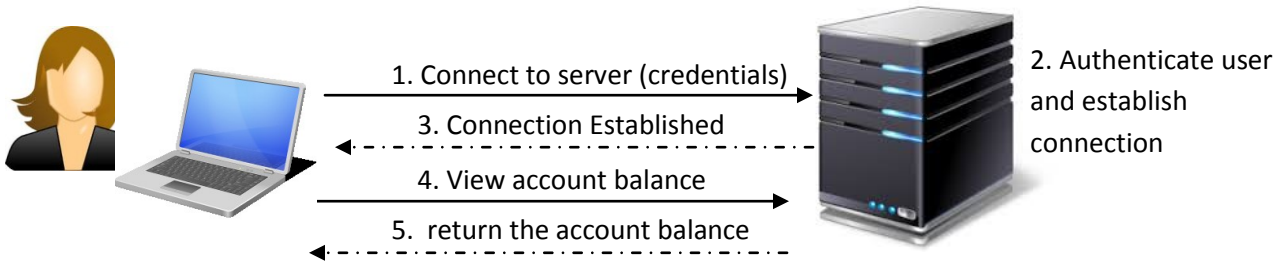
Released: March 11th, 2016

1. Background

At this point in the semester, your group has developed a full understanding of the system to be implemented, congratulations! Your deliverables for this phase of the project will include (i) an SDD (Software Design Document), (ii) an executable prototype of your software program. You need to read this document and the attached SDD document **carefully** before you proceed. Your group's grade for this phase of the project will be based on two aspects: (i) your ability to demonstrate your understanding of the basic Object Oriented Programming concepts that we have been discussing in class and your correctness of applying it to your own project design, (ii) the program prototype that you will hand in.

2. General Project Requirements:

During Phase I of the project, the only general requirement you needed to pay attention for was that your project needs to fit within a client-server model. This means that you are actually developing two separate applications, one which will execute and run as a server and the other as a client. The server application will accept requests from the client application to perform certain functionalities. For example, in a banking application, the server application will be holding in the data for the bank accounts and clients. A client side application can send a request to the server application to view the client's account balance. The server application correspondingly will retrieve the client's bank account, lookup the balance and return this information as a response to the client application. For simplicity, we will assume for all projects that only one



instance of a client application and one instance (thread) of the server are executing (i.e. the server is not expected to handle multiple connections to multiple clients simultaneously).

The other general requirement for the remaining phases of the project will focus on demonstrating the use of most of the OOP concepts that we have been discussing in class. That being said, each project needs to demonstrate the use of the following OOP design concepts and some other Java tools:

a- Concrete classes and objects.
b- Composition and Inheritance relationships
c- Abstract classes and Interfaces
d- Java Collections Classes
e- Exception Handling
f- Sockets Programming
g- File I/O for objects serialization
h- Collections

We have not covered all of these topics yet in class, but we will get to it one at a time. Therefore, for now you do not need to worry about implementing all of these concepts in your code, you just need to show at this phase of the project how you can take advantage of what we have covered so far: mapping real world problem to Classes/Objects that use both composition and inheritance relationships.

2. What do you need to do?

Task 1: Design your system

Modular programming is defined as the process of subdividing a computer program into separate sub-programs. A module is a separate software component. It can often be used in a variety of applications and functions with other components of the system. Similar functions are grouped in the same unit of programming code and separate functions are developed as separate units of code so that the code can be reused by other applications.

Object Oriented Programming (OOP) is compatible with the modular programming concept to a large extent. Modular programming enables multiple programmers to divide up the work and debug pieces of the program independently.

This is the time for your group to practice developing modular code, to do so, you need to break your system into modules that can be implemented independently, this will help accelerate the implementation as well as help each group distribute the implementation of the different modules on the group members. Examples of modules could be: FileIO Modules (for reading and writing different data into files), Networking Modules (for handling connections between servers and clients), Input Verification Modules (for verifying users' inputs), Dispatch Controller Module (for controlling the flow of data from one module to the other), and so on. The modules should be implemented in a generic way that they can handle any types of data.

Put well thought and consideration during this phase as a good design can save a lot of time for the implementation. For each module think about the expected inputs and outputs of the module and the different functionalities for each module and how these functions will be arranged in classes and objects.

A portion of your grades will be given on how well you designed your system, and how you were able to break the functionalities of your system into modules that could be reusable by other systems as well.

Task 2: Fill up the SDD template:

You will find the SDD (Software Design Document) template in the blackboard under the Project directory. You are allowed to make some educated assumptions on parts of the system that you will not be implementing but yet you need to provide information about it in the document (for example: the hardware specifications of the system). In the SDD you will define **details** of the system implementation. You need to specify the following:

1- The modules of your software product (inputs, outputs and functions of each module). Details on each and every function in the module should be defined.

2- Data Structures you are using (arrays, linked lists, binary trees, ..etc) with a brief explanation for the need for each particular data structure.
3- Any files storage system. Note that you will be providing persistence storage of the data held by our objects using disk serialization. You don't need to worry about the implementation of this part yet, we will study it in details in class.
4- Include class diagram(s) to show your class definitions (data members, methods), the relationships between your classes (inheritance or composition), any interfaces or abstract classes used as well. The class diagram should include only those classes that are developed within your program (i.e. do not show Java library classes). You can use any specialized UML modeling tool to draw your class diagram (e.g. Microsoft Visio, IBM Rational Rose .. etc). Note that you will probably need a class diagram for the server application and one for the client application. For a comprehensive review of class diagrams and UML modeling refer to the IBM documentation found here: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/

In the Development section (Section 4) of the SDD you need to specify which modules will be implemented during Phase II of the project and which modules will be implemented as part of the final product of this project.

Important note: In the SDD make sure you clearly show the distinction between the classes and objects that will be implemented within the client application and those implemented in the server application.

Task 3: Build a prototype:

Using the software design you set in Task 2, and the general guidelines for the project, you should be able to start implementing the backbone structure of your program. This means that you will transform your class diagram to the skeleton code of your project that will show all the classes and the relationships between them and the methods that will be implemented within each class. At this phase of the project, you do not need to provide complete implementation of all the methods (remember we have not discussed all the concepts yet). But you should be able to implement the simple ones, such as taking user inputs, initializing data members and objects,

mutators and accessors, etc.. Some UML modeling tools (such as Rational Rose) can actually generate the backbone code of your project from the class diagram.

4. Deliverables

For this phase you need to turn in two things by **Wednesday March 30th, 2016**

1- **The SDD documentation**. You need to submit your document in the specified blackboard submission directory AND print out a copy and hand it to me in class. Each group should report the individual effort of each one in the team by listing down which task/subtasks each group member completed.

2- **The software prototype**. Bundle all your implementation in a folder having your project name. Within this folder provide all your source code and a README file that lists the team members' names and any special instructions on compiling and running the program. Also list the things that you have implemented in the prototype and the things that are still under implementation. Any folder submitted without the README file will NOT be graded.

3- Note that from the deadline of Phase II, each group has only four more weeks to finish implementing the entire project and setting it up for a live demonstration for the final grading process.